Realtime File System Replication On FreeBSD

This article describes a concept on how to implement realtime file system replication on a dual-node FreeBSD cluster to provide real HA services.
Maybe you are familiar with DRBD (distributed replicated block device) from the Linux world already, which basically does something we could call network-RAID1.

Since DRBD does not run on FreeBSD one might be tempted to believe that realtime file system replication would not be possible at all. This is not true however. FreeBSD provides you with two valuable geom classes which will allow you to implement a very similar setup: ggate and gmirror.

Requirements

The absolute minimum requirements for this setup are as follows:

- two hardware nodes running FreeBSD
- ethernet connection between both nodes
- a free (as in "unused") disk slice on each node

All right, this is just good enough to get it going.
If you are serious in useing it you may want to stick to something better than that:

- use FreeBSD 6.x whenever possible, 5.x has some serious locking issues
- Don't use the same ethernet connection for public access AND replication, use a dedicated interface instead, preferrably over Gigabit ethernet. We're talking about data replication over a LAN here, so latency and network load is a concern after all.
- Fore the same reasons as above you should not do any geographic separation, especially not over slow links or VPN. Stay within the same network segment.
- Use identical hardware for both nodes.
- Use identical disk partition and slice setup on both nodes.
- Use fast disks and fast disk controllers with good IO performance.
- Refrein from useing geom/ataraid or other software RAID on partitions/slices mirrored to the second node. Use a real hardware RAID controller instead. If you don't, deadlocks may occur.
- Keep the partitions to be mirrored as small as possible. The reason for this is the fact that a complete resync is required if the mirror brakes. While a 20 GB partition might synchronize within ~30 minutes across a 100 Mbit network, a 500 GB partition will take over 11 hours.
- You should propably not export more than one disk slice to a remote node. Every request (especially with lots and lots of write transactions) will be sent over your network. This causes load and latency on both nodes.

Pros

- Build a two-node HA cluster useing FreeBSD
- Implement realtime file system replication for mission critical failover scenarios
- Use commodity hardware, no need for special shared storage like SAN or iSCSI
- Do not rely on snapshot-based synchronisation (like rsync for example)
- Do not rely on NFS or other file servers which could impose a single point of failure on their own

Cons

- Yet experimental, not tested under heavy-load, possibly unstable

- No support, if it brakes you're on your own
- Implementation not as mature as DRBD
- Yet, a lot of hand work involved

#1 General System Setup

I have already pointed out some recommendations about the system setup previously. So if you stick with these you may save yourself from trouble.

When you install FreeBSD make sure you take a current 6.x series release. The 5.x series might work too though happened to be a bit flacky at my site due to locking issues. YMMV.

There are no special considerations except for the partition layout: reserve a partition which shall contain the data to be replicated to the remote-host. Don't make it to big as the whole thing has to be synchronized over the network.

Choose the size according to your actual disk space requirement, the network speed and latency and also the IO performance of your system. A 500 GB partition may be too big, even when running over Gigabit ethernet. A size anywhere from 100 megs to 20 gigs may be ok though.
Since you would hopefully have two identical nodes, make the partition tables/disk slices match each other. This will help greatly to reduce any issues because of different device names.
You should also refrain from useing any geom/ataraid software RAID on the disks/slices to be exported. Remember that you will do a software RAID1 over the network already. Placeing another software RAID onto the underlying device will lead to deadlocks in most cases. Also your system will have twice the load as the data has to be written out four times actually.
If you really want the additional safety of local disk RAID do yourself a favor and use a real hardware RAID controller instead. This will even help you in getting good IO performance. Of course fask disks are a must then.

My setup consisted of two machines with Intel P-III 800 MHz CPU, 1 GB RAM, two 100 mbit network interfaces (one public, one private) and a RAID1 array with 20 GB SCSI disks (I used an ICP Vortex controllers).

This is what my disk slices look like:

/dev/da0s1a / 8 GB
/dev/da0s1b swap 2 GB
/dev/da0s1d [unused] 10 GB

#2 Enable Kernel Modules

Now make sure both nodes support the GEOM mirroring module. Enable it by adding the following line to your /boot/loader.conf:

geom_mirror_load="YES"

Do the same for the GEOM gate module:

geom_gate_load="YES"

If your secure level allows to load kernel modules at runtime you may omit these steps.
Check it like this:
#sysctl kern.securelevel

Any return value other than 0 or -1 denote that kernel modules may not be loaded at runtime. In this case a reboot is required to load the modules. But check out step #3 first.

#3 Configure Network Interfaces

Make sure your network interfaces are configured properly.

Since I have two of them I would use one as public interface and the other as private.

The latter one will be useing private IP addresses according to RFC1918 and is connected to the remote host useing a crossover cable.

On both hosts fxp0 is the public interface (which later on use the address 172.16.100.1 for the master node and 172.16.100.2 for the failover node).

On the master node the additional public IP address 172.16.100.12 is bound as an alias and used to provide public services. It will be monitored by freevrrpd and conditionally move over to the failover node.

fxp1 is the private interface used for data replication (192.168.100.1 for the master node and 192.168.100.2 for the failover node).

Restart networking or reboot the machine (if required by step #2), whatever applies to you.

#4 Install Failover Software

On FreeBSD freevrrpd may be used for IP takeovers and optional script execution. Install it from the ports (/usr/ports/net/freevrrpd/) or as a binary package (pkg_add -r freevrrpd).

The configuration of the failover setup is fairly easy and well documented in the freevrrpd man page. An example might look like this:

```
#
# config for usual master server
#
[VRID]
serverid = 1
interface = fxp0
priority = 255 # denotes priority = master
addr = 172.16.100.12/32 # denotes failover IP
password = anyoneulike
masterscript = /usr/local/bin/become_master
backupscript = /usr/local/bin/become_standby
```

And this would be an example for a standby node:

```
#
# config for usual standby server
#
[VRID]
serverid = 1
interface = fxp0
```

priority = 240 # denotes priority = failover
addr = 172.16.100.12/32 # denotes failover IP
password = anyoneulike
masterscript = /usr/local/bin/become_master
backupscript = /usr/local/bin/become_standby
Now I'd stronly recommend to read the man page and change the config file according to your needs. You will also need to write the master and backup scripts which do the actions required for the failover to work properly.

I leave this up to you as this is beyond the scope of this howto.

#5 Export Disk Slices

Now export the slices which shall be used for replication (/dev/da0s1d in my case). You do this by creating a file called /etc/gg.exports on the master server:

192.168.0.2 RW /dev/da0s1d

And the same on the standby server:

192.168.0.1 RW /dev/da0s1d

You'll find more on this in the ggated man page. Basically you're just exporting the underlying device to the given IP address in read/write mode.

Now since ggated does not support any password protection or encryption at all it is best to use a dedicated network for this anyway. This will also lower the load you place on the public network segment.
For optimum performance Gigabit ethernet is recommended.

When you're set with the config files, ggated must be started on the failover node (yes: the failover node, not on the master!). You do this by running:

#ggated -v

This will place ggated in verbose mode and run in foreground, which is useful for debugging purposes. Later on, when everything works fine, this can be omitted.

Please note that you should not export the partion on both nodes at the same time. Run ggated only on the host which is the current failover node. Use the freevrrpd master/backup scripts to start/stop the service as required.

#6 Import Disk Slices

Looking at the primary node, the remote disk slices must no be imported.

This is done through ggatec, the client component of ggated. Run it as follows:

#ggatec create 192.168.100.2 /dev/da0s1d

This command will return the device node name. If it is the first one created usally 'ggate0'.

Consider that you should run ggatec only on the designated primary node. Use the freevrrpd master/backup script facilities to create/delete the ggate device node according to it's state.

Do not create the device node on the failover node as long as it is not in primary state. Do not delete the device node as long as the host is in master state (except for recovery purpose, but this will be covered later).

#7 Setup Replication

Now it's actually time to bring up replication. This is where gmirror kernel module enabled previously comes in handy.

Make sure you're on the primary node, then initialize a new GEOM mirror:

#gmirror label -v -n -b prefer gm0 /dev/ggate0

Then insert the local disk slice:

#gmirror insert -p 100 gm0 /dev/ad0s1e

Rebuild the mirror:

#gmirror rebuild rm0 ggate0

If you want to use the geom mirror auto synchronisation features, you can enable these as follows:

#gmirror configure -a gm0

This will cause the disk slices to be synchronized, actually the data from the local ad0s1e will be copied over to the ggate0 remote device.

This will surely take some time, depending on the size of your partition and the speed of your network. When finished, a message like this will appear in the dmesg log of your primary node:

GEOM_MIRROR: Device gm0: rebuilding provider ggate0 finished.
GEOM_MIRROR: Device gm0: provider ggate0 activated.

You may have noticed the "prefer" balance algorithm. This setting actually means that read requests shall only be directed to the geom provider with the highest priority.

By adding the /dev/ad0s1e (which is always the local disk) with a priority of 100 (actually any priority highter then the one of ggate0 according to "gmirror list gm0" output is fine) you force all read requests to be directed to this device only.

You could actually use the "round-robin" balance algorithm as well, however this requires fast network connection with low latency, otherwise your read performance will drop significantly.

You may now "newfs" your gm0 device, mount and use it as you would with any other data partition.

In the first place you should now test the setup. Monitor the system performance on both hosts by using "vmstat" or a similar tool. Keep an eye on network interface and IO statistics.

If you experience lags, timeouts or slowisch behaviour during usual actions like copying files and directories then the above will certainly help you. In most cases it's related to network bandwidth or limits in disk IO.

#8 Failing-Over To The Standby Node

Now that your replication is up and running it's time to test a failover scenario. We do it by hand so you can see what you actually need to put in freevrrpd master/backup scripts for this purpose.

So go and unplug your current master node (yes, really do it. If you don't do it now you'll never do it and it is likely to never work properly).

So you unplugged it? Fine, that's what we want.
Now connect to your failover node and stop the ggated service.

This should cause geom mirror to pick up the gm0 device with provider /dev/da0s1e automatically.

GEOM_MIRROR: Device gm0 created (id=2381431211).
GEOM_MIRROR: Device gm0: provider ad0s1e detected.
GEOM_MIRROR: Force device gm0 start due to timeout.
GEOM_MIRROR: Device gm0: provider ad0s1e activated.
GEOM_MIRROR: Device gm0: provider mirror/gm0 launched.

It may take a few moments for the device to become ready.

Now you must run fsck to ensure filesystem integrity (you really must do this as the filesystem will always be dirty):

#fsck -t ufs /dev/mirror/gm0

Then you can mount the device:

#mount /dev/mirror/gm0 /mnt

Step #9 will explain how the mirror may be rebuilt if the previous master node becomes available again.

#9 Recovering

To bring back the master host into the active combound you will need to make sure that the gm0 device is actually shut down on the failed host.

You remember that we enabled permanent loading of the geom mirror module previously?
This is required to circumvent some problematic situations when kernel secure level is in effect. But it also means that geom mirror will automatically pick up the gm0 device. This will however prevent you from exporting the underlying device through geom gate so the gm0 must disabled first. You can do it like this:
#gmirror stop gm0

As soon as it is stopped you may then run ggated to export the partiton (we're doing it in debug mode):

#ggated -v

If you get an error stating failure to open the /dev/da0s1e device it may still be locked by the geom mirror class. Just look at "gmirror list" output and stop the device as required.
If ggated is running after all, turn to your failover host and turn off auto configuration on the geom mirror:

#gmirror configure -n gm0

Then make the ggate device available to your node:

#ggatec create 192.168.100.2 /dev/da0s1e

Reinsert the ggate device to the geom mirror using a low priority of '0'
#gmirror insert -p 0 gm0 /dev/ggate

and re-enable auto-configuration on the mirror

#gmirror cinfigure -a gm0

I'd recommend to always rebuild the mirror unless your absolutely sure that no new data has been added to the gm0 device in the meantime.

#gmirror rebuild gm0 ggate0

Make sure you give the ggate0 device as last argument which makes it the "sync target". If you happen to do "gmirror rebuild gm0 da0s1e" accidentally this will sync the other way round leaving you most likely with corrupt or lost data.

The rebuild will take some time depending on the partitition size and network speed. After finishing you will see a message like this in your kernel log:

GEOM_MIRROR: Device gm0: rebuilding provider ggate0 finished.
GEOM_MIRROR: Device gm0: provider ggate0 activated.

Now you will have to remove the local /dev/ad0s1d device from the mirror and reinsert it using a high priority:

#gmirror remove gm0 /dev/ad0s1d
#gmirror insert -p 100 gm0 /dev/ad0s1d

The geom mirror will automatically rebuild the provider if required.

This is actually required to fix the read priority I previously talked about, although only required if you want the previous failover node to become your new master node.

If you do not intend switching designated roles and make your failed primary the active node again, have a look at the next sections.

#10 What If The Failover Node Fails?

Imagine you need to reboot your Failover Node, let's say to install some updates. Or even more worse: It has rebooted due to some kernel panic, power loss or other real-life situations.
In any case you should put the geom mirror on the master host into degraded mode by forcibly removing the ggate0 device:

When you're on the master, just make sure the ggate0 is disconnected from the mirror:

#ggatec destroy -f -u 0

This will result in this kernel message:

GEOM_GATE: Device ggate0 destroyed.
GEOM_MIRROR: Device gm0: provider ggate0 disconnected.

The gm0 is now running in degraded state until you re-insert your fail-over node to the configuration.

There is no problem in doing it this way anyway as you have to do a full resync in either case afer the failover node is up again.

The reason to remove the ggate0 device is to prevent IO locking on the geom mirror device.

#11 How To Recover Replication

To bring back the fail-over host into the active combound you will need to make sure that the gm0 device is actually shut down on the failed host.

#gmirror stop gm0

As soon as it is stopped you may then run ggated to export the partiton (we're doing it in debug mode):

#ggated -v

If you get an error stating failure to open the /dev/da0s1e device it may still be locked by the geom mirror class. Just look at "gmirror list" output and stop the device as required.
If ggated is running after all, make the remote disk slice available on the other host:

#ggatec create 192.168.100.2 /dev/da0s1e

This will have the ggate0 device created and added automatically to your gm0 device.

GEOM_MIRROR: Device gm0: provider ggate0 detected.
GEOM_MIRROR: Device gm0: provider ggate0 activated.

I'd recommend to always rebuild the mirror unless your absolutely sure that no new data has been added to the gm0 device in the meantime.

#gmirror rebuild gm0 ggate0

Make sure you give the ggate0 device as last argument which makes it the "sync target". If you happen to do "gmirror rebuild gm0 da0s1e" accidentally this will sync the other way round leaving you most likely with corrupt or lost data.

The rebuild will take some time depending on the partitition size and network speed. After finishing you will see a message like this in your kernel log:

GEOM_MIRROR: Device gm0: rebuilding provider ggate0 finished.
GEOM_MIRROR: Device gm0: provider ggate0 activated.

#12 Data Integrity Considerations

Some special considerations must be taken to ensure data integrity:

- ggated cannot export a slice if it is in use by geom mirror
- don't try any fancy primary-primary replication stuff, it is not possible
- never (as in never) mount the filesystem (the underlying partition to be exact), on the failover node
- to access the data mount the geom mirror device, hence it's only possible on the master node. Don't ever do it on the failover node unless you have taken proper recovery action as described above
- always run fsck on the geom mirror after failover
- it's better not to mount the geom mirror through fstab automatically. Use some freevrrpd recovery magic instead
- Always take backups. This solution is to allow realtime replication for HA services. It is no substitute for proper backups at any time.

#13 Security Considerations

As you may have noticed ggated doesn't support any security or encryption mechanism by default. "Security" is only implemented upon IP based access restrictions combined with read/write or read/only flags.

To enhance security a bit you should always use a dedicated network interface for data replication, preferrably a private one which is not connected to the internet. Crossover host-to-host cabling is fine.

If you need to go over the (insecure) public network please use additonal firewall rules to block port access to authorized hosts.

Both ggated and ggatec also allow useing a port different from their default so it would be possible to setup a redirect through stunnel. This may however pose another performance impact onto your hosts, especially if your network connection is laggy or slow.

#14 Observations

It may look a bit complicated at a first glance, but it is basically nothing else than spanning a software RAID1 accross networked hosts.

In theory its possible to apply any RAID configuration supported by geom accross networked hosts, but there is no practical reason in doing so.

The possibilites offered by this setup are huge if implemented properly. You can easily apply HA conditions to services which do not support such on their own.

If you happen to implement a live environment upon this technology some time, just let me know how it worked out.